Attorney's Docket No.: 082225.P2170

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Ariel Hendel *et al.*

Serial No.: 08/813,647

Filed: March 7, 1997

For: METHOD AND APPARATUS FOR PARALLEL TRUNKING OF INTERFACES TO INCREASE TRANSFER BANDWIDTH

Examiner: Thong Vu

Art Unit: 2756

DECLARATION UNDER 37 CFR § 1.131

Assistant Commissioner for Patents
Washington, D.C. 20231

Dear Sir:

I hereby declare that I am one of the co-inventors of the above-referenced patent application having Serial No.: 08/813,647. I further declare that:

1. I conceived this invention in or before June of 1996 in Mountain View, California with the co-inventors named in the above referenced patent application.

2. A "white paper" entitled "Simple Trunking Model" describing the pertinent features and advantages of the above referenced invention was prepared for internal company distribution on or before August 29, 1996.

3. A true and correct copy of the "white paper" entitled "Simple Trunking Model" prepared for internal company distribution dated August 29, 1996 describing the pertinent features and advantages of the above referenced invention is attached hereto.

4. As of September 1996, a first embodiment of the invention described in the above referenced patent application was implemented.

5. Subsequent to the conception of the invention, I submitted an invention disclosure to the Legal Department of Sun Microsystems, Inc.

6. On information and belief, the Legal Department of Sun Microsystems, Inc. retained the law firm of Blakely, Sokoloff, Taylor & Zafman to prepare a patent application for the above referenced invention which was filed on March 7, 1997.

7. I hereby declare that all statements made herein of my own knowledge are true, that all statements made herein on information and belief are believed to be true, and all statements herein are made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001, Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the above referenced patent application or any patent issued thereon.

Dated: 12/1/00

Leo A. Hejza

Simple Trunking Model (STruM)

Howard Frazier, Leo Hejza, Ariel Hendel
SMCC/INPG
August 29, 1996

1. Introduction

Ethernet has learned a new trick in the past couple of years. It has learned how to scale its link speed by a factor of ten. 10BASE-T begot 100BASE-T, which in turn will be scaled up to one gigabit per second with 1000BASE-T. The Ethernet community has deliberately and emphatically rejected the idea of specifying link speeds in anything other than multiples of 10. Proposals have been made, and shot down in flames, for intermediate link speeds between 10 and 100 Mbps, and they have recently been made again in the context of Gigabit Ethernet, and they have been met with the same reception. No one who sells networks or computers wants to confuse the marketplace by producing multiple, non-interoperable, intermediate link speeds, and everyone seems to feel more comfortable when the link speed can be expressed as power of the number of our fingers or toes.

Never the less, at any given point in time, the choice of link speeds (10, 100, 1000 Mbps) may not match up very well with the amount of sustained throughput that a particular device can support. Virtually all multi-processor servers shipped today can sustain greater than 100 Mbps aggregate network transfer rates. Furthermore, when switches and high performance routers are used to interconnect multiple links of a given speed, there is a clear need for the inter-switch or inter-router link to be able to support at least some aggregation of the links. The next power of ten increase in link speed may not be an attractive choice from a cost standpoint unless the utilization of the higher speed link is going to be greater than 40 to 50 percent. Kicking up the link speed also requires new hardware.

To solve this problem, the concept of "trunking" has long been used in some networks. For the purposes of this discussion, trunking will be defined as the ability to combine multiple parallel physical links into one logical channel. We will limit ourselves to trunks in which the physical links share a common source, and a common destination. We will further limit ourselves to trunks in which each of the links (or "segments" of the trunk) have identical physical layer and media access control layer characteristics. We have paid particular attention to the way IP packets will be transported over these trunks, but we don't believe that the model described herein is in any way limited IP networks.

The remainder of this paper will describe a set of rules that the equipment on each end of the trunk must agree to. The rules can be applied to either end stations (DTEs, such as computers of any classification) or network infrastructure components (specifically switches). The rules are symmetric, which is to say that both ends are subject to the same rules.

2. Rules

1. A trunk may have any number of segments, but all segments must have identical physical layer and media access control layer characteristics
2. Each segment of the trunk shares a common source and a common destination with the other segments of the trunk
3. Temporal ordering of the packets transported across a given segment of the trunk must be preserved throughout the network, subject only to loss due to bit errors
4. Temporal ordering of the packets transported across different segments of the trunk must not be assumed
5. Packets must not be replicated or duplicated across the segments of a trunk. This includes broadcast and multicast packets
6. Broadcast and multicast packets transmitted through a segment of the trunk must not be "echoed" or "looped-back" to the sender over the other segments of the trunk
7. The model assumes full duplex operation at the physical and media access control layers for each segment. Half duplex operation, using CSMA/CD, is neither supported nor desired
8. End stations connected to trunks will associate a single 48 bit IEEE MAC address with all segments of the trunk.
9. Load balancing across the segments is not assumed to be perfect. Each end of the trunk will attempt to load balance across the segments to the best of its ability, subject to all of the forgoing rules

These rules do not addresses the configuration, setup, management, or maintenance of trunks, nor do they address failure detection or recovery. It is assumed that the physical layer will provide some indication if a particular segment of the trunk fails, and that each end of the trunk will monitor whatever status is provided by the physical layer, and take whatever action is deemed appropriate. Configuration and setup are assumed to be performed via manual operations specific to each implementation.

3. Proposals for load balancing

The diagrams in Figure 1 through Figure 3 may assist the reader in understanding the proposals

Figure 1 shows a trunk connection between a server (as a specific example of a DTE) and a switch. The example shows a trunk with 3 segments, labeled A, B, and C. The switch is also connected to several client segments, labeled a, b, c, etc. In this configuration, it is

possible to replace the server with other types of equipment, such as a router, or a high performance workstation, or a printer, to list a few examples.

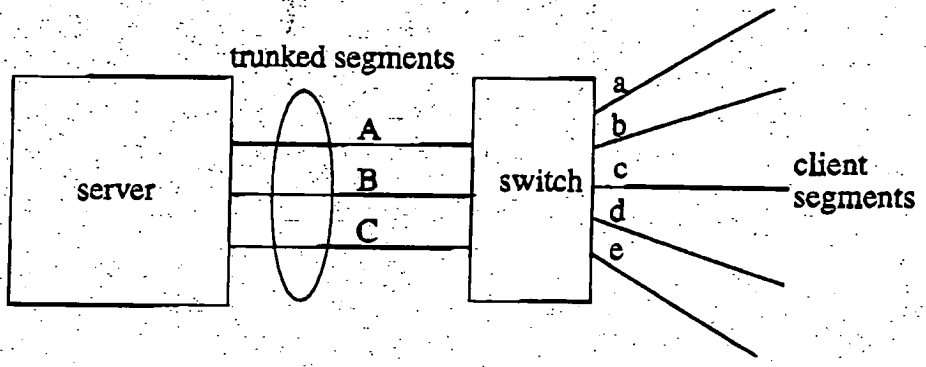

FIGURE 1. Trunks between servers and switches

Figure 2 shows a trunk used as a connection between two switches. As in the previous example, there is no special significance to the number of segments which make up the trunk in Figure 2. The trunk could just as easily be made of two or four or practically any number of segments, depending on the amount of bandwidth required.

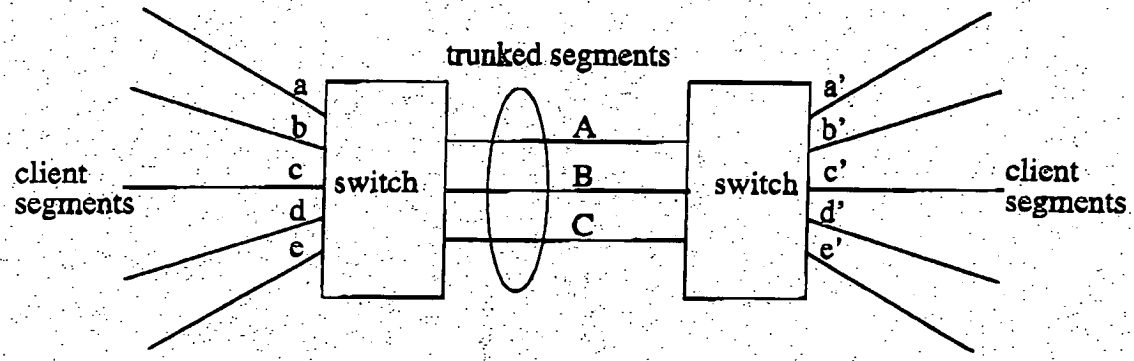

FIGURE 2. Trunks between switches

Figure 3 shows a trunk employed between two servers. Once again, either or both of the servers could be replaced with some other type of equipment, such as a router.

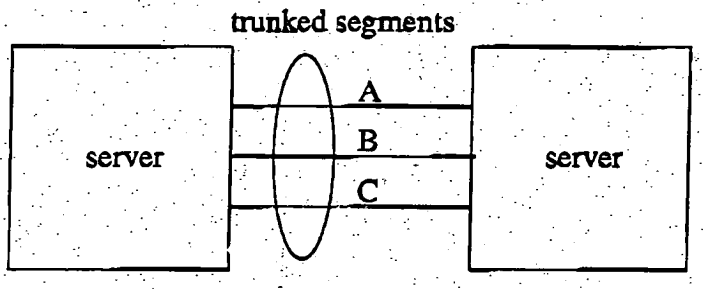

FIGURE 3. Trunks between servers

3.1 Load balancing in servers

An important goal of the model is to make the trunk appear like a single high bandwidth interface from the viewpoint of the server's protocol stack. In addition, all clients which communicate with the server via the trunk should have a consistent view of the server's identity (MAC and IP host addresses). Load balancing by managing the Address Resolution Protocol (ARP) tables in the clients has been considered and rejected because it would dramatically increase the number of ARP frames emitted by the server, and would require a significant amount of added functionality in the server's ARP implementation.

In order to satisfy Rule # 4, "Temporal ordering of the packets transported across different segments of the trunk must not be assumed"the server must ensure that all packets of any sequence of packets which requires temporal ordering are transmitted over the same segment of the trunk.

It should be noted that transport protocols generally can recover from situations where packets arrive out of order, but that this generally entails a significant degradation in throughput, because out of order reception is handled as an exception, and is not optimized. Therefore, the server load balancing mechanism should be designed to take advantage of Rule # 3, "Temporal ordering of the packets transported across a given segment of the trunk must be preserved throughout the network, subject only to loss due to bit errors".

At this point, it might be helpful to introduce a diagram which shows the software components of a trunked server interface.

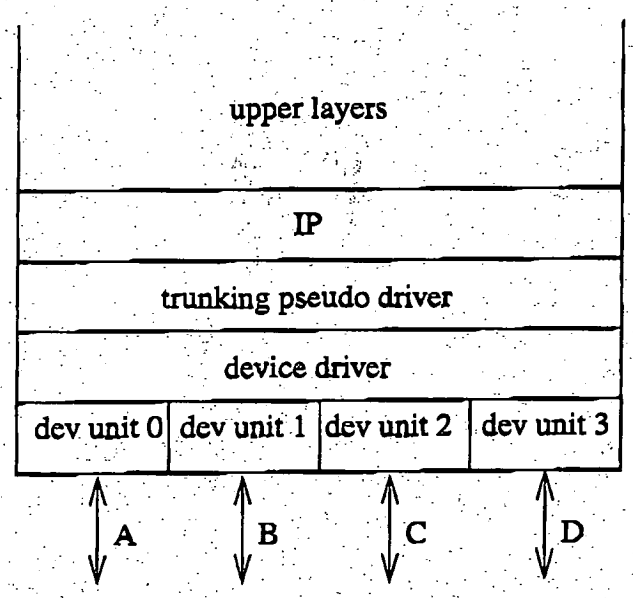

FIGURE 4. Software components of a trunked server interface

A "trunking pseudo driver" is introduced between the IP protocol layer and the network device driver. The function of the pseudo driver is to act as a demultiplexor in the transmit path, and a multiplexor in the receive path. In order to satisfy the rules regarding temporal ordering, the pseudo driver will attempt to ensure that all of the packets associated with a

particular transport layer datagram are enqueued on the same network device transmit queue. This assumption is made on the basis that ordering within a datagram is sufficient, and that ordering between datagrams is unnecessary.

It would be quite difficult for the pseudo driver code to inspect the headers of each packet and attempt to associate them with a particular datagram, though one can imagine that this could be accomplished given an arbitrarily fast processor to execute the code. As a first approximation, the authors feel that it would be sufficient to keep a small cache of MAC (or IP) destination addresses associated with each network interface (each segment of the trunk). Thus, when IP hands the pseudo driver a packet, the pseudo driver checks the cache to see if it has recently transmitted a packet to this DA. If it has, the pseudo driver will enqueue the packet on the same interface that it enqueued the last packet to this DA. If this DA has not been transmitted to recently, the pseudo driver can either enqueue the packet on the least busy transmit queue (the emptiest queue), or the next available queue in a round robin fashion. What ever queue it selects, the driver must update the cache for that queue with the new DA.

In the degenerate case in which a server is talking to one and only one client, this technique would ensure that all packets to that client travel over the same interface, and hence the same segment of the trunk. Why do we call this load balancing? Because it works much better in a non-degenerate case, and will do a good job of ensuring ordered delivery if we can safely assume that the degree of interleaving of packets to different DAs between IP and the network driver is of the same order as the number of processors in a given server. As a first guess, the depth of the cache should be equal to roughly twice the number of processors in a given server. The deeper the cache, the more casual the updating to the cache can be. Experimentation to derive the optimal value for the depth of the cache, and to explore the trade-offs between caching layer 2 and layer 3 addresses is warranted.

3.2 Load balancing in switches

Several switch load balancing mechanisms are possible for forwarding packets into a trunk. The set of load balancing guidelines listed below apply to both switch to switch and switch to server trunks. They ensure that the switch behavior is consistent with conventional bridging guidelines.

1. No frame misordering for a given priority level between a given MAC source and destination.
2. No frame duplication.
3. Transparent to protocols operating above the MAC layer

A natural approach to load balancing is to emulate a faster link by keeping all trunk segments equally busy, possibly by using the corresponding output queue as the metric for how busy the segment is. As long as the links implement flow control, the output queue length is a good end to end proxy for the segment utilization (without flow control, a high segment load is not necessarily reflected in the state of the output queue due to packet loss on the receive queue at the other end).

Deciding for every packet which segment to use, based solely on queue length, might lead to the frame misordering prohibited by the first guideline. If the decision is only a function of the source address of the packet, or of the packet's port of arrival then the first guideline is always satisfied. This scheme however results in a static load balancing function, and the trunk effectiveness depends on the distribution of the traffic sources. While we anticipate that large number of traffic sources would result in acceptably even distributions, it is still possible to end up with configurations where the mapping function forwards most of the traffic to the same segment.

To handle these cases, It is possible to have a dynamic mapping function and still maintain frame ordering, as long as the function changes are slower than the output queue transit times. For instance, the mapping for a given source address can be determined at the time the first packet with the source address is seen, and eventually aged when the source address is not seen for a period of time.

Having the mapping function consider both the source address and the port of arrival reduces the number pathological cases. For example if the traffic is spatially dominated by a particular input port, considering the source address helps spread its traffic, and conversely the port of arrival helps distribute traffic dominated by a small number of addresses (servers or routers) in particular if more than one trunk exists in the switch.

Prevention of frame duplication is achieved by treating the set of trunked ports as if they were a single port, with a separate queue per segment, and making sure that all forwarding is done to only one of its queues.

Furthermore, for the purposes of other 802.1d functions like learning MAC addresses, filtering frames, and executing the Spanning Tree Protocol (if applicable) trunked ports are also treated as if they were a single port.

So far the discussion was centered around using MAC layer information for load balancing. It is possible for the switch to observe higher level protocol information in order to make better load balancing decisions, as long as the third rule of protocol transparency is followed. Transparency implies that the protocols are not aware nor explicitly cooperate with the switch load balancing function. In addition, for protocols that are not supported or understood by the switch, connectivity must be still guaranteed.

Load balancing based on higher level information is practical for switches that examine Layer 3 headers on a packet by packet basis. Many switches examine Layer 3 headers once, for VLAN configuration for example, and use the corresponding Layer 2 information for packet processing. The potential load balancing merits of this approach were not considered.

3.2.1 Other Approaches

The aim of the mapping function could also be different than equally balancing the segments. For example it could separate traffic according to priority, or whether the traffic is bandwidth managed or best effort. A priority based approach is supported by the first

guideline, because packet order preservation is not necessary across different priorities. A priority based approach is straightforward whenever the priority information is well defined at the MAC level (for example if VLAN tags are used).

Restating the main observations, we have shown that the switch behavior is conceptually simple, guided by a set of simple rules along with the particular switch architecture, and can be defined independently of the server load balancing behavior.